# Perspective-aware texture analysis and synthesis

Weiming Dong, Ning Zhou, Jean-Claude Paul

**The Visual Computer manuscript No.**
(will be inserted by the editor)

**Weiming Dong · Ning Zhou · Jean-Claude Paul**

# Perspective-aware texture analysis and synthesis

**Abstract** This paper presents a novel texture synthesis scheme for anisotropic 2D textures based on perspective feature analysis and energy optimization. Given an example texture, the synthesis process starts with analyzing the texel (TEXture ELement) scale variations to obtain the perspective map (scale map). Feature mask and simple user-assisted scale extraction operations including slant and tilt angles assignment and scale value editing are applied. The scale map represents the global variations of the texel scales in the sample texture. Then, we extend 2D texture optimization techniques to synthesize these kinds of perspectively featured textures. The non-parametric texture optimization approach is integrated with histogram matching, which forces the global statics of the texel scale variations of the synthesized texture to match those of the example. We also demonstrate that our method is well-suited for image completion of a perspectively featured texture region in a digital photo.

**Keywords** Texture synthesis · Perspectively-featured texture · Scale map

## 1 Introduction

Texture synthesis is defined in [1] as "a texture synthesis method starts from a sample image and attempts to produce a texture with a visual appearance similar to that sample".

The visual appearance of an example can be analyzed by the *local* continuity of the texels(TEXture ELement), and *global* perspective features for example texel scale variation. These features are due to environmental changes, e.g. perspective view-point, luminance, object distributions and geometry of the underlying surface.

Traditional approaches analyze local properties of a given example and create visually similar images by comparing local neighborhoods. Therefore, they could nicely preserve the *local* continuities of the texels. Nevertheless, these approaches are not sensitive to the global perspective features and hence limited to relatively isotropic examples.

In this paper, we present a new energy optimization-based texture synthesis algorithm for *perspectively featured textures* (PFTs). The proposed algorithm has the ability to extract the perspective scale variations of the texels in the example, and hence preserve the example visual property in the output results. As shown in Fig. 1, from an example texture, our technique can inherit the global texel scale variations in the output, either from an automatic method (the sub-brunch using feature mask) or interactive method (the sub-brunch using user-given slant and tilt angles). The main contributions of our work consist of the three following aspects:

- A novel scheme for synthesizing a large variety of PFTs by integrating the techniques of 1) texture synthesis, 2) shape-from-texture, 3) interactive image editing, and 4) energy optimization.
- An effective texel scale evaluation method based on the texture feature mask and user-given slant and tilt angles.
- A new energy optimization based algorithm for synthesizing textures with perspective variations.

In the rest of the paper, we first introduce the related work on texture synthesis and shape-from-texture in Sec. 2. Then, in Sec. 3, we discuss the details of our texel scale evaluation methods, with feature mask or user-given slant and tilt angles. The extension of the texture optimization algorithm for synthesizing PFTs is described in Sec. 4. After showing the experimental results in Sec. 5, we conclude the paper and show some directions for future work in Sec. 6.

Weiming Dong
LIAMA-NLPR, CAS Institute of Automation, Beijing, China
E-mail: wmlake@gmail.com

Ning Zhou
Tsinghua University, Beijing, China
E-mail: zhoun03@mails.tsinghua.edu.cn

Jean-Claude Paul
Tsinghua University, Beijing, China/INRIA, France
E-mail: paul@tsinghua.edu.cn

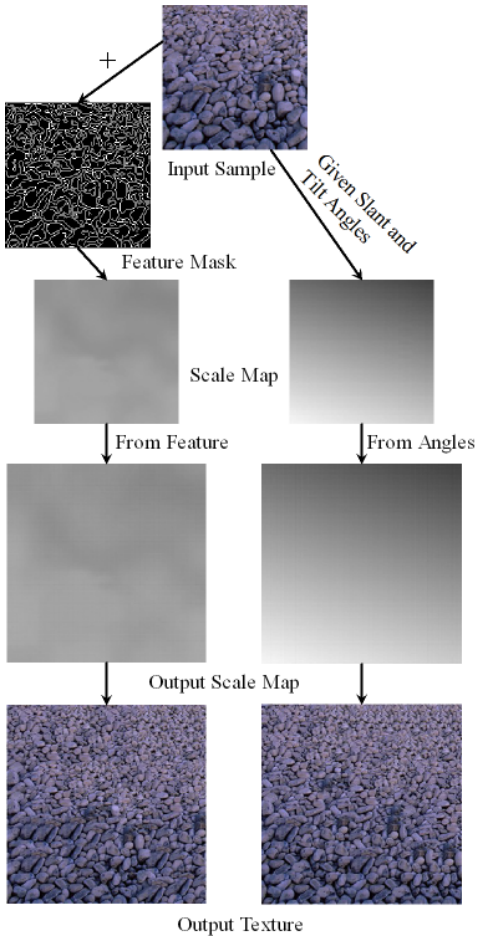



**Fig. 1** Our perspective-aware texture synthesis algorithm. The scale map could be generated in two ways: extracted from the feature mask, or directly calculated from the user-given slant and tilt angles.

## 2 Related Work

A number of works has been presented on synthesizing 2D textures from input examples. Local region-growing techniques generate textures one pixel or one patch at a time. Pixel-based synthesis algorithms [9,22,1,10] grow an output texture pixel by pixel, normally using spatial neighborhood comparison to match across different frequency bands. These approaches are fit for stochastic textures, but usually fail on textures with more coherent structures. Patch-based methods [16,8,13,24,26,12] copy selected source regions into the output instead of single pixels. Recently, efficient GPU-based texture synthesis techniques [14,15] have also been proposed. Another category is to pre-compute correlative tiles [3,7] or similarity sets [25] for runtime references. They usually achieve real-time performance while sacrificing some result qualities. Above techniques always assume the input examples as isotropic (while most of them do not strictly match this demand), so some global features of the inputs could be lost if no special treatment is employed during the synthesis process.

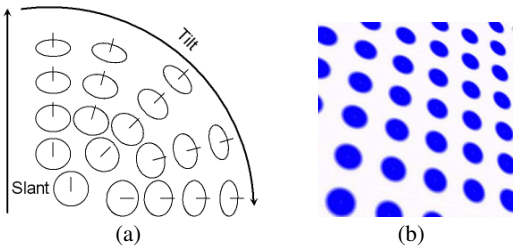
Existing work on analyzing and manipulating input samples has yielded impressive results for texture synthesis. Dischler et al. [5] decompose textures into elementary components and recompose similar textures by taking into account the previously computed arrangements. Liu et al. [18] develop a multi-modal framework to define and capture deformation fields from near-regular textures. Using their formulation, simple parametric models could be constructed from input texture samples to purposefully manipulate the regularity of near-regular textures. The techniques of synthesizing and editing structured textures have also been presented [6]. They assimilate texture images to corresponding 2D geometric meshes, while synthesis is then based on the creation of new vertex/polygon distributions matching some arrangement map. However, all the techniques will have difficulties in getting a feasible lattice or mesh when the texels are too complex to be segmented, or even when there is no clear texel shape in the example, especially for some natural textures [1]. Our method does not require the segmentation of texels, but use some very simple user assistance instead to evaluate the exemplar perspective features.

To preserve the texel scale variations in output images (Fig. 1), we need to extract the scale field from the example. This approach is concurrent with the research work in shape-from-texture. A wide variety of sophisticated shape-from-texture algorithms exist for recovering the shape and orientation of a surface through measuring the distortion of the texels on it [17,20,2,23]. These methods usually treat shape-from-texture as a statistical estimation problem and measure relative metric changes between the surface and the image plane. The results are 3D coordinates of the surface or slant and tilt angle for a plane. The texel scales are also evaluated during the recovery process.

## 3 Texel scale evaluation

In our approach, we assume that the texture image captured from the real 3D surface is locally planar, so that the texture variations are only produced by the projective geometry. We do not deal with the geometric deformation as in [18] for near-regular textures. The same method could be employed if necessary. For visually plausible texture synthesis, we do not need to find a strictly consistent scale and transform angles. Instead we will evaluate some visual properties of the texture to compute the relative scale differences among the texture elements, and perform texture synthesis according to the constraint of the scale map.

The texel scale variation in a texture is directly affected by the surface orientation [19]. Here we represent surface orientation using a viewer centered spherical coordinate system that is parameterized in terms of *slant* ($\sigma$) and *tile* ($\tau$). We consider the possible optical projections of a circular disk at varying orientations relative to the line of sight (see Fig. 2(a)). The optical projection of a circle is always an ellipse. The slant of the circle in 3D space determines the



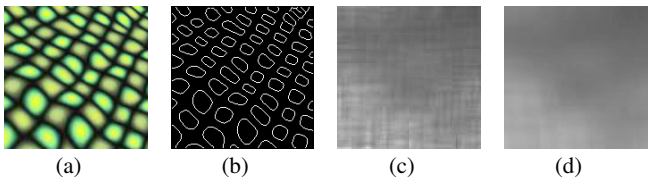

(a) (b)

**Fig. 2** Illustration of texture scale variation. (a) A set of circular patches arranged on a sphere to illustrate the slant and tilt components of surface orientation. The line at the center of each patch is aligned in the direction of the surface normal. Note that the slant and tilt components are from a spherical coordinate system, in which lines of latitude have constant slant, and lines of longitude have constant tilt. (b) A circle texture image with $slant = 45°$ and $tilt = 30°$.

aspect ratio of its projected ellipse, whereas the tilt component determines the orientation of the ellipse within the image plane. We are able to see the shape and scale variations of different circles in one image caused by the perspective projection in Fig. 2(b). Note that in this paper, we use a 90°-rotated version for the slant definition in [19]. We call these kinds of textures which possess perspective deformation features perspectively featured textures (PFTs).

### 3.1 Scale recovery from feature mask

For any PFT $t_p$, there exists an underlying mapping principle $M_{sca}$ that illustrates the scale distortion of each point from an isotropic texture $t_i$. Clerc and Mallat [2] give the formulations for recovering the normals and projective angles from a textured image through the texture gradient equations. With these pieces of information the scales could be evaluated. Unfortunately, their algorithm is very complicated and difficult to be implemented in small time. And we find the following feature mask-based model works very well with many examples to calculate the scale maps.



(a) (b) (c) (d)

**Fig. 3** Extract scale map through binary feature mask. (a) Input texture. (b) Binary feature mask. (c) Original noisy scale map. (d) Final smoothed scale map.

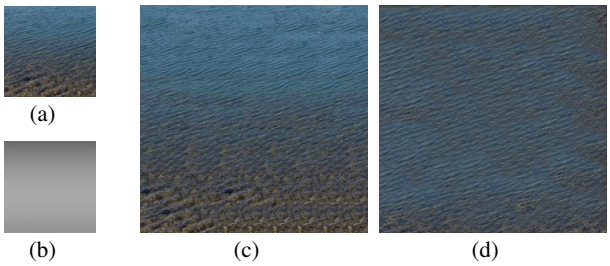
Wu and Yu [24] introduce the notion of feature mask to help guide the synthesis process. Their idea can be simply used in our scheme. Given a binary feature mask of the input texture $t_p$ (Fig. 3(b)), for each point $p$, we compute the quantity of the feature pixels in the mask within its user-assigned $n \times n$ neighborhoods ($n = 32$ for Fig. 3(c)). We denote the point with minimum feature pixel count $a_{min}$ as the largest scaled point, while the point holding maximum $a_{max}$ feature pixels has the smallest scale. We consider the point with the medium count $a_{mid} = (a_{min} + a_{max})/2$ as the non-distorted point which has the scale of 1. Then the scale $S(p)$ of each point is calculated as

$$S(p) = a(p)/a_{mid} \tag{1}$$

where $a(p)$ is the number of feature pixels within the neighborhood of $p$. Now we can get a very noisy scale map through Equation (1) (Fig. 3(c)). The reason is that the neighborhoods of some points might not contain the proper quantity of feature pixels that can be mapped to its real scale. We use a simple but very effective method to smooth this map. For each point in the scale map, we calculate the average scale value $S_{avg}(p)$ and the middle scale value $S_{mid}(p)$ in its neighborhood, and set the scale value of $p$ as

$$S(p) = \alpha \cdot S_{avg}(p) + (1.0 - \alpha) \cdot S_{mid}(p) \tag{2}$$

where $\alpha \in (0,1)$. In this paper, we set $\alpha = 0.8$. The result can be seen in Fig. 3(d), we smooth the scale map in a $9 \times 9$ neighborhood for each point. Usually the user could adjust the size of the neighborhood according to the noisy degree of the original map. Note that we visualize the scale map as a gray image. A gray value of 128 means the original scale ($S(p) = 1$), and the increase of white value means a magnification on the original texel scale.



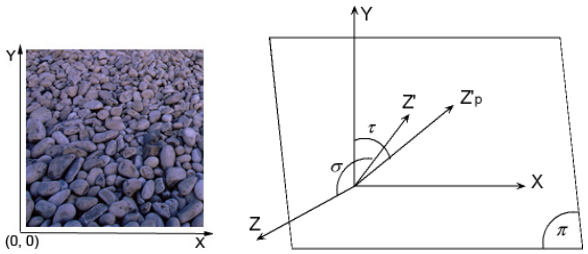
(a) (b) (c) (d)

**Fig. 4** Calculate texel scales from given slant and tilt angles. (a) Input texture. (b) Scale map calculated from given projective angles with $slant = 46°$ and $tilt = 0°$. (c) Synthesis result with our algorithm. (d) Synthesis result with texture optimization [12].

### 3.2 Scale recovery from angles

The feature-mask-based texel scale evaluation method could not get feasible scale maps for some textures. For example in Fig. 4, the texel shapes of the sea texture are blurred when the waves are far from the viewpoint, so it is difficult to *automatically* compute the scale map through the statistical model in Sec. 3.1.

On the assumption that the texel distortion of a locally planar texture image is only due to the perspective projection, we build an approximate scale map generation algorithm




(a) Image coordinate (b) Slant and tilt representation

**Fig. 5** Geometric illustration for recovering scale map from projective angles. In (b), $XY$: the image plane; $Z$: the normal of image plane; $\pi$: the texture plane; $Z'$: the normal of texture plane; $Z'_p$: the projection of $Z'$ on $XY$; $\sigma$: the slant angle (angle between $Z$ and $Z'$); $\tau$: the tilt angle (angle between $Y$ and $Z'_p$)

(a) (b) (c) (d)

**Fig. 6** Calculate texel scales from given slant and tilt angles. (a) Input texture. (b) Binary feature mask. (c) Scale map calculated from the feature mask. (d) Scale map calculated from given projective angles with $slant = 60^\circ$ and $tilt = 18^\circ$.

based on the user-provided slant and tilt angles. As shown in Fig. 5(a), we set the origin of the image plane to the left-bottom corner of the input sample. And we always transform the texel with the largest scale to the origin while the smallest is on the right-top, through mirroring or flipping operations. Fig. 5(b) shows the definitions of the slant angle and tilt angle in 3D space. They are the only required user-input parameters in our algorithm.

We assume that the maximum and minimum scale values are determined only by the slant angle $\sigma$, which can be calculated as

$$S_{max} = 1.0/\cos\sigma,\ S_{min} = \cos\sigma,\ S_{dif} = S_{max} - S_{min}$$

where $S_{max}$ is the maximum scale value and $S_{min}$ is the minimum one, and $S_{dif}$ is the difference. Let $(w,h)$ indicate the width and height of the texture image, then the point coordinates on the image plane are varying from $(0,0)$ to $(w-1,h-1)$. We compute the maximum $Y$-coordinate value $Y_{max}$ and the minimum one $Y_{min}$ of the image plane points on the texture plane as

$$\begin{aligned} Y_{max} &= (w-1-w/2)\cdot\sin\tau + (h-1-h/2)\cdot\cos\tau \\ Y_{min} &= -Y_{max} \\ Y_{dif} &= Y_{max} - Y_{min} \end{aligned} \tag{3}$$

where $Y_{dif}$ is the difference value between $Y_{max}$ and $Y_{min}$. Apparently Equation (3) is based on the assumption that the focus of the camera is on the center of the texture plane. Then the local scale $S(p)$ of each point on the image plane is specified by

$$\begin{aligned} p(Y_{proj}) &= (p(x)-w/2)\cdot\sin\tau + (p(y)-h/2)\cdot\cos\tau \\ S(p) &= Y_{max} - (p(Y_{proj}) - Y_{min}) * S_{dif}/Y_{dif} \end{aligned} \tag{4}$$

where $p(x)$ and $p(y)$ are the coordinates of image point $p$, $p(Y_{proj})$ evaluates the $Y$-coordinate value of an image point when it is projected back onto the texture plane. The scale map computed for the texture image in Fig. 4(a) is shown in Fig. 4(b). We always get a smooth scale map with this method. By comparison of the scale maps generated by feature mask and perspective angles in Fig. 6, we could see that for the texture with a clear texel shape, the scale map created from the feature mask could better represent the real texel scale variations in the example. Compared with previous work on shape-from-texture in [20], our method does not require the camera local length and the distance between the inclined plane and the camera to be known.

### 3.3 Interactive Scale Editing

The scale feature field recovered from the feature mask of a texture or user-provided projective angles is sometimes unsatisfying due to various reasons: random texel shape formation, non-uniform texel distributions and different texture types etc... Moreover, the lighting conditions (shadows, caustics) sometimes also affect the texel appearances. Instead of attempting to handle these complications automatically, we develop several intuitive tools for tuning the result interactively.

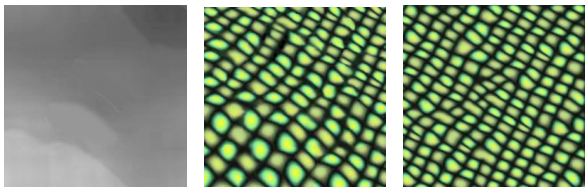
Visualizing the scale feature field as a gray-valued image, the user can magnify or minify the scale values by increasing white or black values of the pixels. The user can also duplicate some values from one area to another. Finally, the variation of scales over a region can be manipulated, as demonstrated in Fig. 7.

(a) (b) (c) (d)

**Fig. 7** Points value variations of the scale feature field in Fig. 3(d) is magnified (a), minified (b) and duplicated (c). (d) is the final scale map after editing.

## 4 Scale-preserved optimization

We include the scale value as an additional image channel in the texture synthesis process. To produce a similar global



(a) Output scale map (b) Our Result (c) Result of [12]

**Fig. 8** Re-sample the input scale map (Fig. 7(d)).

appearance in the synthesized image as input, we first re-sample the input feature maps according to the required output size (Fig. 8(a)). The re-sampled scale map is set to be the initial value of the scale channel of the output image. The scale-preserved optimization process for synthesizing PFTs begins with an image where the value of each point is randomly chosen according to the scale channel, i.e. the point is chosen from the points which have the same scale value in the input example.

### 4.1 Optimization phase

Denoting by $e$ the input example, and by $x$ the synthesized texture, the global texture energy that we seek to minimize is defined as

$$E_t(x;e) = \sum_p \|x_p - e_p\|^r \tag{5}$$

Here $x_p$ and $e_p$ refer to the neighborhoods centered around pixels. In the texture optimization algorithm, $e_p$ is the closest neighborhood to $x_p$ (in $L_2$ norm). The exponent $r = 0.8$ causes the optimization to be more robust against outliers.

We use iteratively re-weighted least squares (IRLS), similarly to [12] and [11], to minimize the energy. To this end, we rewrite the terms of the energy functional (5) as follows:

$$\|x_p - e_p\|^r = \|x_p - e_p\|^{r-2}\|x_p - e_p\|^2 = \omega_p\|x_p - e_p\|^2 \tag{6}$$

and minimize the following quadratic functional:

$$E_t(x;e) = \sum_p \omega_p\|x_p - e_p\|^2 \tag{7}$$

### 4.2 Search phase

In this phase we optimize Equation (5) with respect to $e_p$ by finding the best matching exemplar window for every neighborhood $x_p$. This is a standard nearest neighbor search in a high-dimensional space, and it dominates the running time of our optimization.

We speed this step up in a number of ways. First, we apply a PCA projection to the neighborhood vectors in the example [10,16,15,11]. We keep only the number of coefficients sufficient to preserve 95% of the variance. For RGBS (S represents the scale channel) textures with $8 \times 8$ neighborhoods the dimensionality is usually reduced from 256 to about $15-40$ dimensions, depending on the size and richness of the example. Thus, performance is drastically improved. We can also apply PCA projection separately to the color channels and scale channel, and keep more information of the scale channel. This operation could improve the similarity between the output and the example on the global texel scale variations.

### 4.3 Histogram matching

For some textures the optimization process could converge to a wrong local minimum, because the energy function measures only the similarity of local neighborhoods, without accounting for any global statistics. Especially for our approach of PFT synthesis, we should keep the global perspective feature of the example in the output, so the result should reflect the full richness of the example.

Similarly to [11], we solve this problem by using a re-weighting scheme designed to make certain global statics of the resulting texture remain close to those of the example. More specifically, we carefully adjust the weights in Equation (7) so as to effectively ensure that certain histograms of the synthesized texture match the example. Like the method in [11], we modify the weight for Equation (7) in the following way:

$$\omega_p = \frac{\omega_p}{1 + \sum_{j=1}^{k} \max[0, H_{x,j}(b_j(e_p)) - H_{e,j}(b_j(e_p))]} \tag{8}$$

where $k$ is the number of bins in the histogram, $H_{x,j}$ and $H_{e,j}$ denote the $j$-th histogram of the synthesized result and the example, respectively, and $H(b)$ denotes the value of bin $b$ in a histogram $H$. For a color $c$, $b_j(c)$ specifies the bin containing $c$ in the histograms $H_{x,j}$ and $H_{e,j}$.

Histogram matching causes the *global* statistics of the texture to match the example, while the neighborhood matching terms of the optimization enforce *local* similarities. The integrated approach automatically adapts itself to the current situation: If the synthesis histograms are far off the example ones we effectively prevent bad texels from contributing to the synthesized value, whereas if the histograms are close, the weights are largely unaffected and the synthesis turns to neighborhood-matching only.

## 5 Results and Discussions

The perspective-aware texture synthesis results which use scale maps extracted from feature masks are shown in Fig. 1, Fig. 8(b) and Fig. 9. Compared with the corresponding texture optimization [12] results, our images successfully preserve the global texel scale variation trends. Fig. 8 and Fig. 10 show the synthesis results from the use of edited scale maps.

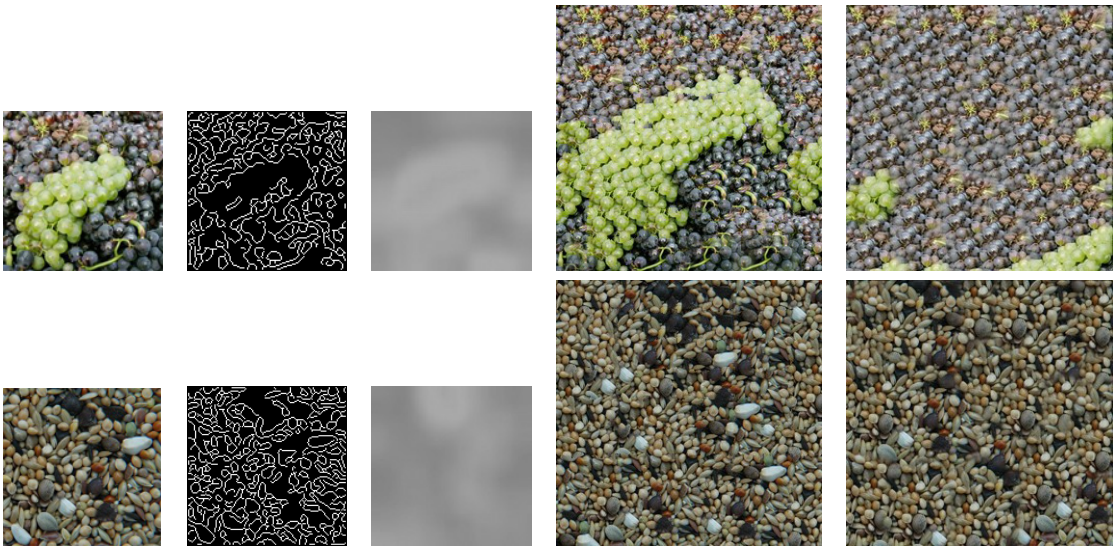


**Fig. 9** Perspective-aware texture synthesis which could preserve the texel scale variations of the input examples in the results. For each row, from left to right: the input example, the feature mask, the scale map, our result, texture optimization [12] result.

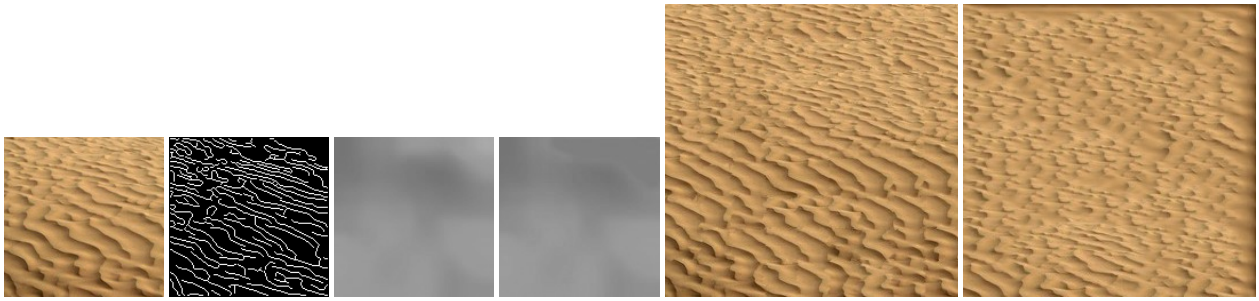


**Fig. 10** Perspective-aware texture synthesis. From left to right: the input example, the feature mask, the scale map, the scale map after interactive scale editing, our result, texture optimization [12] result.

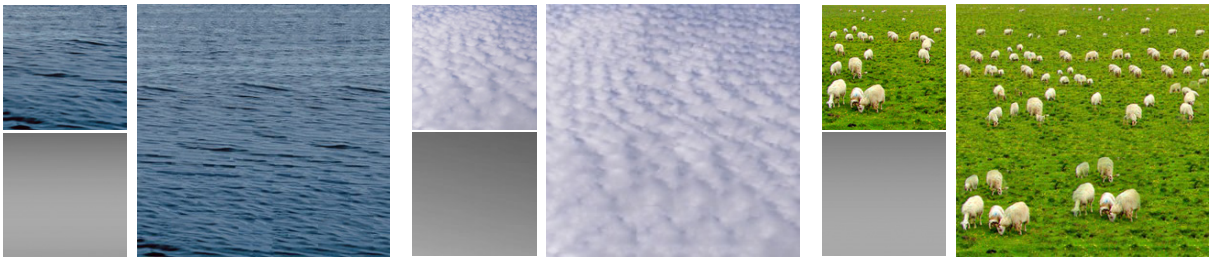
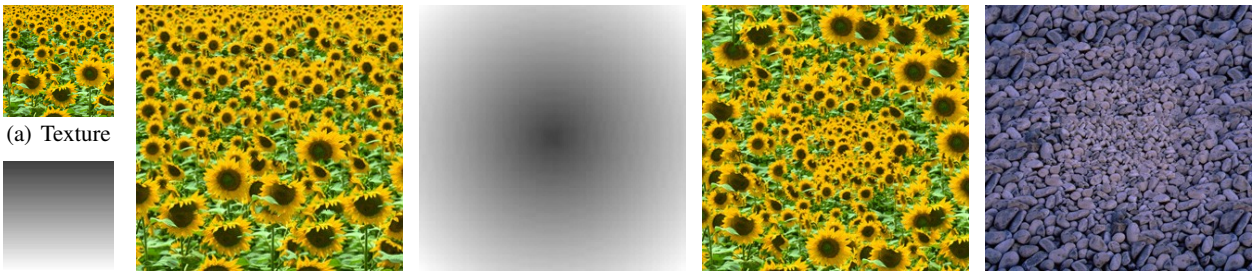


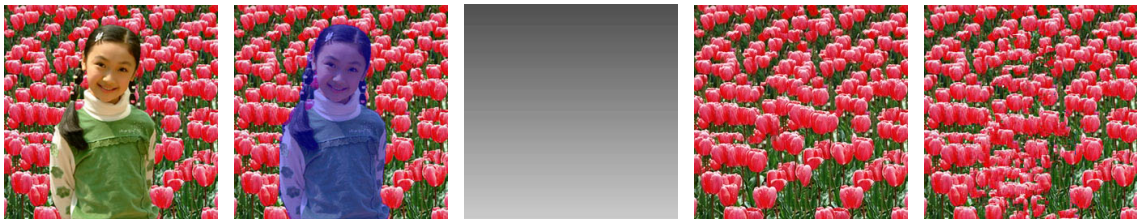
**Fig. 11** Perspective-aware texture synthesis. The scale maps are generated according to user-given slant and tilt angles. For all the groups, from left to right, the slant and tilt angles are: ($slant = 45^\circ, tilt = 5^\circ$); ($slant = 38^\circ, tilt = 10^\circ$); ($slant = 55^\circ, tilt = 0^\circ$).

**Fig. 12** Comparison of perspective-aware texture synthesis results using different user-given slant and tilt angles. From left to right: the example, our result of ($slant = 30^\circ, tilt = 5^\circ$), our result of ($slant = 45^\circ, tilt = 5^\circ$), our result of ($slant = 60^\circ, tilt = 5^\circ$), the texture optimization result.



(a) Texture (b) Scale (c) Scale-preserved synthesis (d) User-specified output map (e) User-specified synthesis (f) User-specified synthesis

**Fig. 13** Scale-aware texture synthesis. (f) is generated from the stones texture example in Fig. 1.



(a) Input image (b) Unknown region (c) Scale map (d) Our result (e) EBI result

**Fig. 14** Image completion for textured region by perspective-aware synthesis.

In Fig. 11 we show some perspective-aware texture synthesis results using scale maps generated from user-given slant and tilt angles. Note that for these examples, it is difficult to extract feasible scale maps directly from the feature masks. We compare the results using different input scale maps in Fig. 12. We can see that the appearance of the output result could be controlled by setting different slant and tilt angles. We show user-specified perspective-aware synthesis results in Fig. 13(e) and Fig. 13(f), we synthesize these two images by following the given output scale pattern in Fig. 13(d).

Compared with previous works which can generate results with similar appearances, such as [13], [26] and [21], our technique is more flexible and effective. In [13], the graph-cut method could only expand the example in one direction, the output image must have the same height or same width as the input. While with our method, we can generate arbitrary size of output images. In [26], the input must be isotropic or treated as isotropic (no scale variations among the texels), so the concept of their scheme is totally different to our work. For the texture design approach in [21], the result is generated by hand and could not assure to exactly keep the global appearance of the input example. On the other hand, the approaches which simply use color value as the reference for user-controlled image synthesis [1, 10] are also not fit for the synthesis of PFTs, especially for the examples without apparent color variations, like the sand texture in Fig. 10 and the sea texture and cloud texture in Fig. 11.

We have extended our perspective-aware texture synthesis technique to image completion where textured regions are required as completed. As shown in Fig. 14, we recover the scale map from the background and then synthesize the unknown region by treating the other area to be the example texture. Compared with the Exemplar-Based Inpainting (EBI) method [4], our method successfully preserves the texel scale variations in the resulting image.

## 6 Conclusion and Future Work

We have presented a method that allows performing perspective-aware texture synthesis for examples with perspectively global appearances. Our method is guided by scale maps that roughly mask the perspective features in the texture images; the synthesis results produced by our algorithm then preserve the appearances of those features or present user-defined feature constraints. These results would be difficult or at least cumbersome to achieve with current software.

Automatic feature analysis for a given PFT is a very complicated problem. In our current implementation, a little simple user assistance is employed to help to extract feasible scale maps, including (1) scale map editing; (2) slant and tilt angle adjustment for texel scale estimation. The perspective-aware technique is also useful in many applications involving texture synthesis, for example the image completion approach in this paper.

In our future work, we plan to add the information of texel directions and texel color variations to the synthesis process. And how to extract a scale map from non-planar textures is also a challenging problem.

**Acknowledgements** This work is supported by National Natural Science Foundation of China projects No. 60073007, 60473110; by National High-Tech Research and Development Plan 863 of China under

Grant No. 2006AA01Z301; and by the MOST International collaboration project No. 2007DFC10740.

## References


1. Ashikhmin, M.: Synthesizing natural textures. In: SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics, pp. 217–226. ACM Press, New York, NY, USA (2001)
2. Clerc, M., Mallat, S.: The texture gradient equation for recovering shape from texture. IEEE Trans. Pattern Anal. Mach. Intell. **24**(4), 536–549 (2002)
3. Cohen, M.F., Shade, J., Hiller, S., Deussen, O.: Wang tiles for image and texture generation. ACM Trans. Graph. **22**(3), 287–294 (2003)
4. Criminisi, A., Peréz, P., Toyama, K.: Object removal by exemplar-based inpainting. In: CVPR '03: Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03), vol. 2, pp. II–721–II–728 (2003)
5. Dischler, J.M., Maritaud, K., Lévy, B., Ghazanfarpour, D.: Texture particles. Computer Graphics Forum **21**(3), 401–401 (2002)
6. Dischler, J.M., Zara, F.: Real-time structured texture synthesis and editing using image-mesh analogies. Vis. Comput. **22**(9), 926–935 (2006)
7. Dong, W., Zhou, N., Paul, J.C.: Optimized tile-based texture synthesis. In: GI '07: Proceedings of Graphics Interface 2007, pp. 249–256. ACM, New York, NY, USA (2007)
8. Efros, A.A., Freeman, W.T.: Image quilting for texture synthesis and transfer. In: SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 341–346. ACM Press, New York, NY, USA (2001)
9. Efros, A.A., Leung, T.K.: Texture synthesis by non-parametric sampling. In: ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2, p. 1033. IEEE Computer Society, Washington, DC, USA (1999)
10. Hertzmann, A., Jacobs, C.E., Oliver, N., Curless, B., Salesin, D.H.: Image analogies. In: SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 327–340. ACM Press, New York, NY, USA (2001)
11. Kopf, J., Fu, C.W., Cohen-Or, D., Deussen, O., Lischinski, D., Wong, T.T.: Solid texture synthesis from 2d exemplars. In: SIGGRAPH '07: ACM SIGGRAPH 2007 papers, p. 2. ACM, New York, NY, USA (2007)
12. Kwatra, V., Essa, I., Bobick, A., Kwatra, N.: Texture optimization for example-based synthesis. ACM Trans. Graph. **24**(3), 795–802 (2005)
13. Kwatra, V., Schödl, A., Essa, I., Turk, G., Bobick, A.: Graphcut textures: image and video synthesis using graph cuts. ACM Trans. Graph. **22**(3), 277–286 (2003)
14. Lefebvre, S., Hoppe, H.: Parallel controllable texture synthesis. ACM Trans. Graph. **24**(3), 777–786 (2005)
15. Lefebvre, S., Hoppe, H.: Appearance-space texture synthesis. ACM Trans. Graph. **25**(3), 541–548 (2006)
16. Liang, L., Liu, C., Xu, Y.Q., Guo, B., Shum, H.Y.: Real-time texture synthesis by patch-based sampling. ACM Trans. Graph. **20**(3), 127–150 (2001)
17. Lindeberg, T., Garding, J.: Shape from texture from a multi-scale perspective. Computer Vision, 1993. Proceedings., Fourth International Conference on pp. 683–691 (11-14 May 1993). DOI 10.1109/ICCV.1993.378146
18. Liu, Y., Lin, W.C., Hays, J.: Near-regular texture analysis and manipulation. ACM Trans. Graph. **23**(3), 368–376 (2004)
19. Norman, J.F., Todd, J.T., Norman, H.F., Clayton, A.M., McBride, T.R.: Visual discrimination of local surface structure: Slant, tilt, and curvedness. Vision Research **46**, 1057–1069 (2006)
20. Plantier, J., Lelandais, S., Boutte, L.: A shape from texture method based on local scales extraction: precision and results. In: Proceedings of International Conference on Image Processing, pp. 421–435. IEEE Computer Society, Washington, DC, USA (2001)
21. Shen, J., Jin, X., Mao, X., Feng, J.: Deformation-based interactive texture design using energy optimization. The Visual Computer **23**(9-11), 631–639 (2007)
22. Wei, L.Y., Levoy, M.: Fast texture synthesis using tree-structured vector quantization. In: SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp. 479–488. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (2000)
23. White, R., Forsyth, D.A.: Combining cues: Shape from shading and texture. In: CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 1809–1816. IEEE Computer Society, Washington, DC, USA (2006)
24. Wu, Q., Yu, Y.: Feature matching and deformation for texture synthesis. ACM Trans. Graph. **23**(3), 364–367 (2004)
25. Zelinka, S., Garland, M.: Jump map-based interactive texture synthesis. ACM Trans. Graph. **23**(4), 930–962 (2004)
26. Zhang, J., Zhou, K., Velho, L., Guo, B., Shum, H.Y.: Synthesis of progressively-variant textures on arbitrary surfaces. In: SIGGRAPH '03: ACM SIGGRAPH 2003 Papers, pp. 295–302. ACM, New York, NY, USA (2003)


**Weiming Dong** is a postdoc at the Sino French Lab in Computer Science, Automation and Applied Mathematics (LIAMA). He received his BSc and MSc degrees in computer science in 2001 and 2004, both from Tsinghua University, P. R. China. He received his PhD in computer science from the University of Henri Poincaré Nancy 1, France, in 2007. His research interests include texture synthesis, computational photography and realist rendering.

**Ning Zhou** is a PhD candidate of the Department of Computer Science and Technology, Tsinghua University, P. R. China. She received her BSc in 2003 in computer science from Tsinghua University. Her research interests include texture synthesis, computational photography and botany-based modeling.

**Jean-Claude Paul** is a Director of Research INRIA and a Professor at Tsinghua University. He received his PhD in Mathematics from the University of Paris and was also graduated in Architecture Design from the ENSBA in 1976. In 1995, Prof. Jean-Claude Paul obtained the Academie des Sciences Prize and the Academie des Beaux Arts Prize, for both his artistic and scientific works. His research interests include realistic rendering, geometry processing and curves and surfaces theory.